

01-14-00

A

Please type a plus sign (+) inside this box [+]

PTO/SB/05 (12/97)

Approved for use through 09/30/00. OMB 0651-0032

Patent and Trademark Office: U.S. DEPARTMENT OF COMMERCE

Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

UTILITY PATENT APPLICATION TRANSMITTAL

(Only for new nonprovisional applications under 37 CFR 1.53(b))

Attorney Docket No. 04411.P002Total Pages 2First Named Inventor or Application Identifier Roni KorenshteinExpress Mail Label No. EL236239606US

je135-U-S-PTO

09/484609

01/18/00

ADDRESS TO: Assistant Commissioner for Patents
Box Patent Application
Washington, D. C. 20231

APPLICATION ELEMENTS

See MPEP chapter 600 concerning utility patent application contents.

1. X Fee Transmittal Form
(Submit an original, and a duplicate for fee processing)
2. X Specification (Total Pages 37)
(preferred arrangement set forth below)
 - Descriptive Title of the Invention
 - Cross References to Related Applications
 - Statement Regarding Fed sponsored R & D
 - Reference to Microfiche Appendix
 - Background of the Invention
 - Brief Summary of the Invention
 - Brief Description of the Drawings (if filed)
 - Detailed Description
 - Claims
 - Abstract of the Disclosure
3. X Drawings(s) (35 USC 113) (Total Sheets 16)
4. X Oath or Declaration (Total Pages 4)
 - a. Newly Executed (Original or Copy)
 - b. Copy from a Prior Application (37 CFR 1.63(d))
(for Continuation/Divisional with Box 17 completed) (**Note Box 5 below**)
 - i. DELETIONS OF INVENTOR(S) Signed statement attached deleting inventor(s) named in the prior application, see 37 CFR 1.63(d)(2) and 1.33(b).
5. Incorporation By Reference (useable if Box 4b is checked)
The entire disclosure of the prior application, from which a copy of the oath or declaration is supplied under Box 4b, is considered as being part of the disclosure of the accompanying application and is hereby incorporated by reference therein.
6. Microfiche Computer Program (Appendix)

01/18/00

je135-U-S-PTO

01/18/00

je135-U-S-PTO

7. _____ Nucleotide and/or Amino Acid Sequence Submission
(if applicable, all necessary)
a. _____ Computer Readable Copy
b. _____ Paper Copy (identical to computer copy)
c. _____ Statement verifying identity of above copies

ACCOMPANYING APPLICATION PARTS

8. _____ Assignment Papers (cover sheet & documents(s))
9. _____ a. 37 CFR 3.73(b) Statement (where there is an assignee)
_____ X b. Power of Attorney
10. _____ English Translation Document (if applicable)
11. _____ a. Information Disclosure Statement (IDS)/PTO-1449
_____ b. Copies of IDS Citations
12. _____ Preliminary Amendment
13. X _____ Return Receipt Postcard (MPEP 503) (Should be specifically itemized)
14. _____ a. Small Entity Statement(s)
_____ b. Statement filed in prior application, Status still proper and desired
15. _____ Certified Copy of Priority Document(s) (if foreign priority is claimed)
16. X _____ Other: Certificate of Express Mail with copy of postcard showing contents of
Express Mail package.

17. **If a CONTINUING APPLICATION**, check appropriate box and supply the requisite information:
_____ Continuation _____ Divisional _____ Continuation-in-part (CIP)
of prior application No: _____

18. Correspondence Address

_____ Customer Number or Bar Code Label _____
(Insert Customer No. or Attach Bar Code Label here)
or

X _____ Correspondence Address Below

NAME Ronald C. Card Ronald C. Card 1/18/2000
BLAKELY, SOKOLOFF, TAYLOR & ZAFMAN LLP

ADDRESS 12400 Wilshire Boulevard
Seventh Floor

CITY Los Angeles STATE California ZIP CODE 90025-1026

Country U.S.A. TELEPHONE (408) 720-8598 FAX (408) 720-9397

UNITED STATES PATENT APPLICATION
FOR

CACHING OUTPUT FROM AN OBJECT IN AN APPLICATION SERVER ENVIRONMENT

INVENTORS:

RONI KORENSHTEIN
RAJESH NARAYANASWAMY

PREPARED BY:

BLAKELY, SOKOLOFF, TAYLOR & ZAFMAN, LLP
12400 WILSHIRE BOULEVARD
SEVENTH FLOOR
LOS ANGELES, CA 90025-1026

(408) 720-8598

EXPRESS MAIL® MAILING LABEL NUMBER

96236239606 US

DATE OF DEPOSIT

1/18/00

I HEREBY CERTIFY THAT THIS PAPER OR FEE IS BEING DEPOSITED
WITH THE UNITED STATES POSTAL SERVICE "EXPRESS MAIL POST
OFFICE TO ADDRESSEE" SERVICE UNDER 37CFR 1.10 ON THE DATE
INDICATED ABOVE AND IS ADDRESSED TO THE ASSISTANT
COMMISSIONER FOR PATENTS, WASHINGTON, D.C. 20231

JUANITA BRUCE

(TYPED OR PRINTED NAME OF PERSON MAILING PAPER OR FEE)

Juanita Bruce

(SIGNATURE OF PERSON MAILING PAPER OR FEE)

CACHING OUTPUT FROM AN OBJECT IN AN APPLICATION SERVER

ENVIRONMENT

FIELD OF THE INVENTION

The present invention relates to data caching and more particular to data caching
5 for streamed output from component objects in an application server environment.

BACKGROUND OF THE INVENTION

A request sent from a web client (such as a web browser) is received by a web
server. The web server then passes the request to an application server which processes
the request, calculates the results, and streams the results back to the web server for
display. Web/application servers cache output as a function of the input (parameters in
the request). These servers typically cache an entire page of data.

Data is cached so that the processing does not have to be repeated for every
request for the same page of data. When caching is implemented, some processing is
performed and the results are cached for some period of time. Any subsequent requests
15 that match some criteria for the cached data result in the web server streaming the
output that has been stored in the cache rather than to re-calculate it. In large
applications, it is sometimes necessary to cache components contained within the page
rather than the entire page. This allows a page to have components that expire at
different times, such that only these components are recalculated for the next request
20 for that page.

The Netscape Application Server, NAS, (also called KIVA Enterprise Server), of
Netscape Corporation of Mountain View, California, has been used here to compute

and stream out the dynamic components of a Web page. In Kiva, component objects to stream data have been written using Kiva's AppLogic class. Kiva provides an application programming interface (API) that provides a set of programming instructions that accomplish a well-defined, modular task within the application. In order to process multiple components within a page of data, each component makes a new request to the application server. Each new request requires a new thread and generates all the objects to execute, which consumes valuable system resources. KIVA provides a caching technique to cache each request.

A disadvantage of using the Kiva Enterprise Server caching technique is that every new request needs to be sent to the application server. This results in a greater amount of overhead as there is a high cost associated with starting out a new request. Thus, if a given page contains five different components, the server will process this as six separate and full requests (the page plus the first components) and each request would carry with it the high cost of starting out a new request. Also in this solution, the new request which is being cached needs to be specially programmed to activate the caching. This becomes a maintenance problem as every object that uses caching must be examined, in order to modify it.

SUMMARY OF THE INVENTION

A method and system of streaming a page of data are described. In one embodiment, an object corresponding to the page of data is allocated. The object is executed. If the object is a proxy, then the proxy is executed. If the object is a component, then the component is executed. If the object is a container, then the container is executed.

004411.P002

BRIEF DESCRIPTION OF THE DRAWINGS

The present invention is illustrated by way of example, and not by way of limitation in the figures of the accompanying drawings in which like reference numerals refer to similar elements.

5 **Figure 1** is an illustration of one embodiment for a distributed object caching system 100;

Figure 2 is a block diagram of one embodiment for an architecture for a computer system;

Figure 3 is a block diagram of one embodiment for an object caching server;

10 **Figure 4a** is a block diagram of one embodiment for an application server;

Figure 4b is a block diagram of one embodiment for an object application;

Figure 5 is a block diagram of one embodiment for random access memory of the object application of **Figure 4b**;

15 **Figure 6** is a block diagram of one embodiment for a component cache of **Figure 3**;

Figures 7-10 illustrate exemplary user input and modification windows;

Figure 11 is a flow diagram of one embodiment for streaming object data for a page of data invoked by a uniform resource locator (URL);

20 **Figure 12** is a flow diagram of one embodiment for streaming data for a component;

Figure 13 is a flow diagram of one embodiment for streaming data for a container;

Figure 14 is a flow diagram of one embodiment for streaming data for a proxy;
and

Figure 15 is a flow diagram of one embodiment for caching output data from an
object associated with a proxy.

004411.P002

DETAILED DESCRIPTION

In one embodiment, objects within a page of data may be written as Java objects. Objects may be components, proxies, or containers of objects. If the page of data has multiple components, the processing costs associated with each start-up are eliminated by processing the caching of the objects in a single request for the page as a whole. A given request is associated with only one start-up expense and all objects are processed during the execution of the request. In one embodiment, a proxy is created in place of the object which executes like the object. At the time the object is to process and stream out its data, the proxy checks input parameters against a cache criteria for the object. If the cache criteria is satisfied and the data is in the cache, the data in the cache is streamed out to the base agent associated with the proxy without creating or processing the object component.

If the cache criteria is satisfied and the data is not in the cache, the proxy creates the object and executes it via a caching base agent. The caching base agent captures all the output from the object associated with the proxy into a buffer. When the object finishes streaming data to the buffer, the caching base agent saves the data to the cache. The proxy then streams out the cache to the associated base agent for the proxy, which in turn streams it out to the requesting web client. In one embodiment, an object may contain any number of nested sub-objects to any number of levels of nesting. The objects and nested sub-objects may be components, proxies, or containers.

In one embodiment, each component is a generic class and not a Kiva Enterprise Server specific AppLogic. The components are not coded in any special manner in

order that they may be implementation independent. The component caching may be actuated externally to the component without changes to the component objects.

A portion of the disclosure of this patent document contains material which is subject to copyright protection. The copyright owner has no objection to the facsimile reproduction by anyone of the patent document or the patent disclosure, as it appears in the Patent and Trademark Office patent file or records, but otherwise reserves all copyright rights whatsoever.

In the following detailed description, numerous specific details are set forth in order to provide a thorough understanding of the present invention. However, it will be apparent to one skilled in the art that the present invention may be practiced without these specific details. In some instances, well-known structures and devices are shown in block diagram form, rather than in detail, in order to avoid obscuring the present invention.

Some portions of the detailed descriptions which follow are presented in terms of algorithms and symbolic representations of operations on data bits within a computer memory. These algorithmic descriptions and representations are the means used by those skilled in the data processing arts to most effectively convey the substance of their work to others skilled in the art. An algorithm is here, and generally, conceived to be a self-consistent sequence of steps leading to a desired result. The steps are those requiring physical manipulations of physical quantities. Usually, though not necessarily, these quantities take the form of electrical or magnetic signals capable of being stored, transferred, combined, compared, and otherwise manipulated. It has

proven convenient at times, principally for reasons of common usage, to refer to these signals as bits, values, elements, symbols, characters, terms, numbers, or the like.

It should be borne in mind, however, that all of these and similar terms are to be associated with the appropriate physical quantities and are merely convenient labels applied to these quantities. Unless specifically stated otherwise as apparent from the following discussion, it is appreciated that throughout the description, discussions utilizing terms such as "processing" or "computing" or "calculating" or "determining" or "displaying" or the like, refer to the action and processes of a computer system, or similar electronic computing device, that manipulates and transforms data represented as physical (electronic) quantities within the computer system's registers and memories into other data similarly represented as physical quantities within the computer system memories or registers or other such information storage, transmission or display devices.

The present invention also relates to apparatus for performing the operations herein. This apparatus may be specially constructed for the required purposes, or it may comprise a general purpose computer selectively activated or reconfigured by a computer program stored in the computer. Such a computer program may be stored in a computer readable storage medium, such as, but is not limited to, any type of disk including floppy disks, optical disks, CD-ROMs, and magnetic-optical disks, read-only memories (ROMs), random access memories (RAMs), EPROMs, EEPROMs, magnetic or optical cards, or any type of media suitable for storing electronic instructions, and each coupled to a computer system bus.

The algorithms and displays presented herein are not inherently related to any particular computer or other apparatus. Various general purpose systems may be used with programs in accordance with the teachings herein, or it may prove convenient to construct more specialized apparatus to perform the required method steps. The required structure for a variety of these systems will appear from the description below. In addition, the present invention is not described with reference to any particular programming language. It will be appreciated that a variety of programming languages may be used to implement the teachings of the invention as described herein.

Figure 1 is an illustration of one embodiment for a distributed object caching system 100. Referring to **Figure 1**, server 102 is connected to mass storage device 104. Server 102 and mass storage device 104 are connected via wide area network (WAN) 112 to a variety of clients 106 and 108. Wide area network 112 may be connected to any of a variety of clients 106 and 108.

In one embodiment, a user at client 106 requests a page of data represented by a universal resource locator (URL) from server 102. The request is transmitted via WAN 112 to server 102. Server 102 creates an occurrence of a base agent associated with the page of data and processes the execute method of the occurrence of the base class. The execute method of the base class allocates an object, which may be a component, a proxy, or a container. The execute method of the object is then called. If the object is a component, then it calculates the result and streams out the result to the associated base agent. If the object is a proxy, then it checks the cache to see if the request meets the cache criteria. If the criteria is met, and if the result exists in cache, then the proxy

streams out the cache entry to the associated base agent. If entry is not in the cache, then the underlying object associated with the proxy is executed, using a caching base agent. The caching base agent captures all the output of the underlying object and saves the output into the cache. The cache entry is then streamed out to the associated
5 base agent by the proxy which in turn streams it out to the web client. If the object is a container it generates all the objects contained within the container in which each object may be a component, container, or a proxy.

In one embodiment, the output of each object may be cached in mass storage 104. If caching is not required for the class or type of object, then the process is as described above. However, if caching is required for the class of object, the stored data for the object is retrieved from the cache and streamed out. If the data is not in the cache, the object creates a cache entry for the data and streams the output to a buffer within mass storage 104. Once the object completes streaming the output to the buffer, the output is saved into the cache and streamed out.

Figure 2 is a block diagram of one embodiment for an architecture for a computer system 200. Referring to **Figure 2**, CPU 202 is connected via bus 215 to a variety of memory structures and input/output 210. The memory structures may include, for example, read only memory (ROM) 204, random access memory (RAM) 206, and/or non-volatile memory 208. In one embodiment, CPU 202 is also connected
20 via bus 215 to network interface 212. Network interface 212 is used to communicate between computer system 200 and server 102 and a variety of other computer terminals 108. Network interface 212 is connected to the wide area network 112 by any of a

variety of means such as, for example, a telephone connection via modem, a DSL line, or the like. The architecture shown in **Figure 2** may be utilized for either clients 106, 108, server 102, or both.

Figure 3 is a block diagram of one embodiment for an object caching server 300.

Referring to **Figure 3**, caching server 300 includes mass storage 104 and server 102. In one embodiment, server 102 includes web server 305, application server 310, and object application 320. Web server 305, application server 310, and object application 320 may be maintained within a single server 102 as shown or may be maintained within separate servers. Object application 320 has an associated component cache 325. In one embodiment, component cache 325 is maintained within mass storage device 104. In alternate embodiments, component cache 325 may be maintained within a separate storage devices.

A Uniform Request Locator (URL) request for a page of data is received via the web server 305 at application server 310. In one embodiment, application server 310 transfers control to object application 320 by creating an occurrence of a base agent associated with the URL page and processes the execute method of the occurrence of the base agent. The execute method of the base agent determines which object is associated with the URL page and allocates an instance of the object. The execute method of the base agent passes input parameters to the object. In one embodiment, if the object is a container, the instance of the container determines all of the objects contained within the container and allocates an instance of each object.

The execute method of the object is processed in order to calculate and stream output. The container calls the execute method of each of its objects in order that each object calculates and streams out its own data. If the object is a component, its output is streamed to the base agent. If the object being processed is a proxy, the proxy is
5 processed in place of the component. The proxy determines whether the input parameters for the underlying object match specified cache criteria for the type of the object. If the input parameters match the cache criteria, component cache 325 is checked to see if there is an entry within component cache 325 corresponding to the input request. If a non-empty cached entry is found, the data of the entry is streamed out by
10 the proxy to the base agent for the page. Thus, a previous entry matching cache criteria for the object and current arguments is found in the component cache 325 and is streamed out by the proxy without creating or executing the underlying object. If an empty entry is found (another thread is calculating data for this entry) "Data unavailable" is streamed out by the proxy and its execution ends.

15 If no valid entry was found within component cache 325, an entry may be created. In one embodiment, a cache entry is created if the object class is tagged as allowing data to be cached. The proxy reserves an entry within component cache 325 corresponding to the input parameters and object. The proxy requests object application 320 to initiate an embedded execution of a caching base agent for the target
20 of the object to be cached and object application 320 creates an instance of the caching base agent. The instance of the caching base agent executes the object which streams its output to a buffer. When the object's execution completes, the proxy saves the buffer

containing the accumulated output of the object into the reserved entry in component cache 325. After the buffer is saved in component cache 325, data in the entry is streamed to the caching base agent which streams the output to the browser.

Figure 4a is a block diagram of one embodiment for an application server 310.

5 Application server 310 includes executive server unit 405 and application logic processing unit 410. Requests for a page of data invoked by a URL are received from web server 305 at executive server unit 405. Application logic processing unit 410 transfers control for the request of the page of data to object application 320. In addition, application logic processing unit 410 receives the streamed data from object application 320 and transfers the streamed data to web server 305.

Figure 4b is a block diagram of one embodiment for object application 320.

Object application 320 includes base agent processing unit 450, object processing unit 452, proxy processing unit 454, and caching agent processing unit 460. In addition, object application 320 may also include application server random access memory (RAM) 455 and caching buffer 465. A request for a URL page is received by application server 310 which creates an instance of base agent processing unit for the URL page.

Base agent processing unit 450 processes an execute method of the instance of the base agent. Base agent determines which object is associated with the URL page and creates the object for the URL page. In one embodiment in which the object is a container, base agent processes the execute method of the container which creates all the objects associated with the container. After creating all the instances of the objects associated with the container, object processing unit 452 processes the execute method for each of

the objects. Each instance of the object associated with the container calculates its own output to stream out to the base agent processing unit 450. Normally the execute method of the object calculates the output to stream out and calls a stream result method of object processing unit 452. The stream result method is a component of an object which passes the stream data to the associated base agent which passes the stream data to the browser via the application server 310 and web server 305. The implementation of the stream result method within the object obtains a reference to the instance of the underlying base agent created by the application server. Each object of the container streams out data as a function of its input and/or the environment.

An object's output may be cached as an independent cache entry within component cache 325. If an object is to be cached, the container inserts a proxy in place of the object within the processing stream. For any object that may be a candidate for caching, a proxy is created in place of the object. When the execute method of a container is processed, the execute method of each of the objects is processed to stream out its data. When the object that is being represented as a proxy is executed, the execute method of the proxy is processed by proxy processing unit 454. When the proxy's execute method is processed, it first determines if the object's input parameters match cache criteria for the object. If the cache criteria is met, the proxy determines whether an entry already exists in component cache 325 for the object. If a non-empty entry already exists, the proxy streams the data from component cache 325 to the base agent processing unit 450. If an empty entry is already reserved for the object, an error message is sent as output notifying the user that the data for the object is unavailable.

However, if the entry does not exist, a new entry is created in the component cache 325 by the proxy.

The proxy then creates an embedded execution of a caching base agent. The instance of the caching base agent creates an entry in caching buffer 465 to receive the output from the object represented by the proxy. Caching base agent creates and executes the object represented by the proxy. As the object streams data to its caching base agent, caching agent processing unit 460 captures the output from the object and appends it to caching buffer 465. After the execution of the object is complete, caching agent processing unit 460 transfers the information in caching buffer 465 to the previously created entry in component cache 325. Control then returns to proxy processing unit 454 which retrieves the just-created entry in component cache 325 and streams the data to base agent processing unit 450. As the container and all its components stream data to base agent processing unit 450, base agent processing unit 450 streams the data to application server 310 which streams the data to web server 305.

Figure 5 is a block diagram of one embodiment for random access memory (RAM) 455 of object application 320. Referring to **Figure 5**, RAM 455 includes base agent 502, object 504, proxy 506, and caching base agent 508. Base agent 502 streams the output for a page of data to base agent processing unit 450 and creates the object for the page. Object 504 streams the output results for the object (which could be a container) to application server 310 and web server 305. Proxy 506 executes the proxy substituted for an object and calls caching base agent 508. Caching base agent 508 executes object which streams data to caching base agent which streams data to buffer 465. Finally,

when object execution completes, caching base agent transfers the streamed data from buffer 465 to component cache 325.

Figure 6 is a block diagram of one embodiment for component cache 325.

Referring to **Figure 6**, component cache 325 includes a number of component cache

5 entries 605. Each component cache entry 605 includes key 610 and cache value 615. In one embodiment, key 610 is a concatenation of the name of an object being cached and the input parameters relevant to the cache criteria for the particular cache entries 605. The concatenated string is converted to a hash value in order to create key 610. Cache value 615 is the streamed output data created by an instance of a proxy associated with a particular object as it was executing the specific parameter values.

Keys 610 are created by a concatenation of the cache criteria, which specifies the conditions under which caching is performed for an object, and the object name. Cache criteria may be developed for each type or class of object that is susceptible to caching. In one embodiment, in order to activate the object class, information is entered as to the length of time the information is to be maintained in the cache 325 and under what conditions the caching is to take place. An entry may be maintained in the cache for a specific period of time or may be maintained indefinitely. For example, an object may be cached for a set period of time such as 60 seconds. When the object is first executed, the execute method of the object will be processed and its output will be saved in the cache. The key value created for the cached entry will be removed from the cache by a separately running program 60 seconds after the entry has been created. Prior to removal, any other request that matches a specific object and satisfies the cache criteria

will stream the cache value from the cache 325 directly. After the cached entry has been removed, the first request matching the cache criteria will result in a new entry being created in cache 325.

In one embodiment, the following conditions may be specified for a given class
5 object. First, the condition that no special conditions should be satisfied in order for the object to be cached. This results in the object being cached regardless of any values of the input parameters. Second, the argument value condition indicating a parameter and a value for the parameter which specifies that caching is to be performed only when the specified parameter exists in the set of input parameters and its value matches
10 the value specified. If the value specified is a star (*) or any other symbol designated for this purpose, then any input value matches the parameter. Otherwise, the value must be specified and it must match the value of the parameter exactly. For example, if the conditions specified is "input-param,def", then caching would be performed if the set of input parameters contains the parameter whose name is "input-param" and the value of
15 this parameter is "def". In this example, the output will be cached with the key representing the name of the object concatenated with the matching value "def". If the cache criteria used is "param1, *" then caching is performed if the parameter "param1" exists in the set of input parameters. Since this parameter may have different values for different invocations, each invocation looks up the key made by the concatenation of
20 the name of the object and the current value of the parameter "param1" with special symbol such as ";" used as separator. For example, if the first invocation of this object named "sub" specifies "param1=A", this matches the criteria "param1=*" and an entry

matching the key "sub;A" is created in the cache, "sub" is executed and its output is saved to the entry whose key is "sub;A". During the "lifetime" of this entry, if other requests are made for component "sub", one which specifies "param1=B" and the second specifies "param1=A", the first request will result in executing the object and the output saved in an entry whose key is "sub;B" and the second request will result in the output streamed from the entry which is already in the cache whose key is "sub;A".

Another type of condition may specify that caching should be performed if a parameter exists in the set of input parameters whose name is "argument" and contains a value which is one of several possibilities such as "value 1", "value 2", "value 3". Any of the values specified in the condition may match the criteria. Thus, this condition may specify, for example, "shape, triangle, circle", which means that caching will occur when the value of the "shape" parameter is either "triangle" or "circle". If an invocation of this object contains the parameter "shape" with the value of "square", no match is made for this criteria and output will not be cached. In addition, any of the conditions described above may be combined to form a variety of cache criteria.

Figure 7 is an exemplary user input and modification window. Referring to **Figure 7**, window 700 is displayed when a user initially signs onto the "Etrade" site.

Figure 8 illustrates an exemplary user input and modification window 800. Referring to **Figure 8**, window 800 is displayed for the user to enter a user name and password in order to access certain areas of the "Etrade" site.

Figure 9 illustrates an exemplary user input and modification window 900.

Referring to **Figure 9**, window 900 is displayed after the user initially signs on to the site using his or her user name or password and is taken to the home page of the site.

Figure 10 illustrates an exemplary user input and modification window 1000.

- 5 Window 1000 is displayed when the user enters the market section of the Etrade site. Within window 1000, the most active gainers and losers area 1010 is displayed. The most active gainers and losers area 1010 is a component (object) of the page of data as illustrated by **Figure 10**. In addition, the window 1010 also includes a news area 1020. Both news area and most active gainers and losers are independent components of the page of data and are created by the method and system as described above.

Figure 11 is a flow diagram of one embodiment for streaming object data for a page of data invoked by a uniform resource locator (URL). Initially at processing block 1105, an instance of base agent 502 corresponding to the page of data is allocated. The instance of base agent 502 is associated with the page of data. The execute method of base agent 502 is processed.

At processing block 1110, an instance of an object associated with the page of data is allocated for base agent 502. In addition, the object obtains a reference to the base agent which allocated the object. In one embodiment, the object may be a component, a proxy, or a container. The execute method of the instance of the base agent 502 determines which object is associated with the page of data and creates the appropriate object. If a proxy is created, the proxy receives the name of the underlying object for which it acts as a proxy.

At processing block 1120, the execute method of the instance of the object is processed. The object execute method determines the output data to be streamed out to the base agent 502. If the object is a component, the processing blocks of **Figure 12** are processed. If the object is a container, the processing blocks of **Figure 13** are processed.

5 If the object is a proxy, the processing blocks of **Figure 14** are processed.

Figure 12 is a flow diagram of one embodiment for streaming data for a component. The processing blocks of **Figure 12** are executed if the object executed at processing block 1120 of **Figure 11** is a component. Initially at processing block 1205, the component calculates its output.

At processing block 1210, the component streams its output data to its associated base agent. Each instance of the component contains a reference to the associated base agent created in processing block 1105 of **Figure 11**. To stream its output, the component calls a stream result method of that base agent. A stream result method in the base agent streams the output data to the web browser. A stream result method in the caching base agent, on the other hand, captures this output data in a buffer as described in reference to **Figure 15**.

Figure 13 is a flow diagram of one embodiment for streaming data for a container. The processing blocks of **Figure 13** are executed if the object executed at processing block 1120 of **Figure 11** is a container. Initially at processing block 1305, the execute method of the container generates all the objects associated with and contained within the instance of the container. The objects may be components, proxies, or

containers. The execute method of the container determines the objects associated with the container and creates an instance of every object for the container.

At processing block 1310, after creating all the instances for the objects, the container processes the execute method for each of the objects created. The objects may be components, proxies, or containers. If the object generated is a component, the processing blocks of **Figure 12** are processed. If the object generated is a container, the processing blocks of **Figure 13** are processed. If the object generated is a proxy, the processing blocks of **Figure 14** are processed.

Figure 14 is a flow diagram of one embodiment for streaming data for a proxy. Initially at processing block 1405, it is determined whether the input parameters for the proxy match the cache criteria for the underlying object. Every proxy contains the name of its underlying object. If there is no match, processing continues at processing block 1410. If there is a match, processing continues at processing block 1420.

If the cache criteria is not satisfied at processing block 1405, at processing block 1410, the underlying object for the proxy is allocated. In one embodiment, the object may be a component, a proxy, or a container. The execute method of the proxy determines which object is allocated based on the name passed to it in processing block 1110 of **Figure 11**.

At processing block 1412, the execute method of the underlying object is processed. The object execute method determines the output data and streams it out to the associated base agent for the object. If the object is a component, the processing blocks of **Figure 12** are processed. If the object is a container, the processing blocks of

Figure 13 are processed. If the object is a proxy, the processing blocks of **Figure 14** are processed.

If the cache criteria is satisfied at processing block 1405, at processing block 1415, a cache key is constructed from the underlying object name and current parameter values that matched the cache criteria.

At processing block 1420, it is determined if an entry exists for the cache key within component cache 325. If the cache entry exists, processing continues at processing block 1445. If the cache entry does not exist, processing continues at processing block 1425.

At processing block 1425, an entry corresponding to the cache key is created in the component cache 325. In one embodiment, proxy 506 adds a parameter to the current parameters of the current request. This parameter's value contains the key for which an entry has just been created in the cache and reserves an entry in the cache 325.

At processing block 1430, proxy 506 creates an instance of the caching base agent 508. At processing block 1435, the instance of caching base agent 508 is executed for the object associated with the proxy. Processing blocks 1505 through 1520 of Figure 15 are processed to execute the base agent which creates the output data of the object within the component cache 325.

At processing block 1440, the cache value of the entry associated with the object is streamed from component cache 325 to the associated base agent which streams out the data and processing of the flow diagram of **Figure 14** ends.

If at processing block 1420 it is determined that a cache entry already exists for the cache key constructed in processing block 1415, at processing block 1445 it is determined if the cache entry is empty. If the entry is empty, processing continues at processing block 1450. If the entry is not empty, processing continues at processing block 1440.

At processing block 1450, if a blank entry already exists in component cache 325 for the object, the object output is being cached by another thread. In this case, the object streams out "data unavailable" to the associated base agent for the object.

Figure 15 is a flow diagram of one embodiment for caching output data from an object associated with a proxy. Initially at processing block 1505, a buffer is created for capturing output data from the execution of the associated object.

At processing block 1510, an occurrence of the associated object is allocated. In one embodiment, the object may be a component, a proxy, or a container. The execute method of the instance of the base agent 502 associated with the associated object determines which object is allocated. The object is given a reference to this caching base agent.

At processing block 1515, the execute method of the associated object is executed. The object's execute method determines the output data to be streamed out to the associated base agent for the object. As the object streams out its output data to its associated caching base agent, all output for the associated object is captured by the caching base agent and streamed out to the buffer.

At processing block 1520, after the execution of the associated object completes, the contents of the buffer 465 is transferred to the component cache 325 into the entry whose key is represented by one of the parameters. After the transfer of the data from buffer 465 to component cache 325 is complete, processing continues at processing block 1440 of **Figure 14**.

Several embodiments in the implementation of the streaming of a page of data have been described. The specific arrangements and methods described here are merely illustrative of the principles of this invention. Numerous modifications in form and detail may be made by those skilled in the art without departing from the true spirit and scope of the invention. Although this invention has been shown in relation to a particular embodiment, it should not be considered so limited. Rather it is limited only by the appended claims.

CLAIMS

What is claimed is:

- 1 1. A method of streaming a page of data, the method comprising:
2 allocating at least one object corresponding to the page of data; and
3 executing the at least one object, wherein executing comprises,
4 executing a proxy if the at least one object is a proxy,
5 executing a component if the at least one object is a component, and
6 executing a container if the at least one object is a container.
- 1 2. The method of claim 1 further comprising recursively performing allocating and
2 executing the at least one object to process at least one sub-object contained within
3 the at least one object.
- 1 3. The method of claim 1 further comprising:
2 allocating an occurrence of an associated base agent corresponding to the page of
3 data.
- 1 4. The method of claim 3 wherein executing a component further comprises:
2 calculating output data for the occurrence of the component; and
3 streaming out the data to the associated base agent.
- 1 5. The method of claim 4 wherein streaming further comprises calling a stream result
2 method of the associated base agent.

1 6. The method of claim 4 further comprising creating a reference to the associated
2 base agent within the component.

1 7. The method of claim 2 wherein executing the container further comprising:
2 generating at least one container object from the container; and
3 executing the at least one container object, wherein executing comprises,
4 executing a proxy if the at least one container object is a proxy,
5 executing a component if the at least one container object is a component,
6 and
7 executing a container if the at least one container object is a container.

1 8. The method of claim 7 further comprising recursively performing generating and
2 executing the at least one container object to process at least one container sub-object
3 contained within the at least one container object.

1 9. The method of claim 2 wherein executing a proxy further comprises:
2 determining if a cache entry exists for the occurrence of the at least one object;
3 if a cache entry is not found,
4 allocating a new cache entry; and
5 streaming out a cache entry value.

1 10. The method of claim 9 wherein determining further comprises:
2 matching cache criteria for the cache entry;
3 if the cache criteria does not match,
4 allocating an underlying object associated with the proxy, and
5 executing the underlying object; and
6 if the cache criteria matches,
7 constructing a cache key.

1 11. The method of claim 10 wherein constructing a cache key further comprises:
2 matching at least one input parameter against at least one cache criteria entry.

1 12. The method of claim 9 wherein determining further comprises:
2 examining the cache using a cache key.

1 13. The method of claim 9 wherein allocating a new cache entry further comprises:
2 creating a new cache entry;
3 allocating an occurrence of a caching base agent; and
4 executing the caching base agent.

1 14. The method of claim 13 wherein creating the new cache entry further comprises:
2 creating a new key; and
3 reserving the new cache entry corresponding to the new key.

1 15. The method of claim 13 wherein executing the caching base agent further
2 comprises:
3 creating a buffer entry to capture output data;
4 allocating an underlying object associated with the proxy;
5 executing the underlying object to stream out the output data to the buffer entry,
6 wherein executing comprises,
7 executing a proxy if the at least one object is a proxy,
8 executing a component if the at least one object is a component, and
9 executing a container if the at least one object is a container; and
10 transferring the buffer entry to the new cache entry.

1 16. The method of claim 9 wherein determining further comprises:
2 determining if the cache entry is empty; and
3 streaming out an error message if the cache entry is empty.

1 17. The method of claim 1 wherein the at least one object comprises all components
2 within the page of data.

1 18. The method of claim 1 wherein the at least one object is an executable object.

1 19. The method of claim 1 wherein the at least one object is a component, a proxy, or a
2 container.

1

- 1 20. A method of streaming a page of data, the method comprising:
- 2 (a) allocating an occurrence of at least one base agent associated with the page of
- 3 data;
- 4 (b) allocating an occurrence of a container associated with the at least one base
- 5 agent;
- 6 (c) generating at least one object from the occurrence of the container;
- 7 (d) executing the at least one object; and
- 8 (e) streaming output data of the at least one object to the base agent.
- 1 21. The method of claim 20 further comprising recursively performing (a) through (e)
- 2 to process at least one sub-object contained within the at least one object.
- 1 22. The method of claim 21 wherein the at least one object is a component, a proxy, or a
- 2 container.
- 1 23. The method of claim 21 wherein the sub-object is a component, a proxy, or a
- 2 container.

1 24. A system for streaming a page of data, the system comprising:
2 means for allocating at least one object corresponding to the page of data; and
3 means for executing the at least one object, wherein means for executing
4 comprises,
5 means for executing a proxy if the at least one object is a proxy,
6 means for executing a component if the at least one object is a component,
7 and
8 means for executing a container if the at least one object is a container.

1 25. A system for streaming a page of data, the system comprising:
2 means for allocating an occurrence of at least one base agent associated with the
3 page of data;
4 means for allocating an occurrence of a container associated with the at least one
5 base agent;
6 means for generating at least one object from the occurrence of the container;
7 means for executing the at least one object; and
8 means for streaming output data of the at least one object to the base agent.

1 26. An article of manufacture including one or more computer-readable media with
2 executable instructions therein, which, when executed by a processing device causes
3 the processing device to:

4 allocate at least one object corresponding to the page of data; and

5 execute the at least one object, wherein to execute comprises,

6 execute a proxy if the at least one object is a proxy,

7 execute a component if the at least one object is a component, and

8 execute a container if the at least one object is a container.

1
1 27. An article of manufacture including one or more computer-readable media with
2 executable instructions therein, which, when executed by a processing device causes
3 the processing device to:

4 allocate an occurrence of at least one base agent associated with the page of data;

5 allocate an occurrence of a container associated with the at least one base agent;

6 generate at least one object from the occurrence of the container;

7 execute the at least one object; and

8 stream output data of the at least one object to the base agent.

1

1 28. A system for streaming a page of data, the system comprising:
2 a base agent processing unit to allocate at least one object corresponding to the
3 page of data;
4 an object processing unit to execute the at least one object if the at least one object
5 is a component or a container; and
6 a proxy processing unit to execute the at least one object if the at least one object
7 is a proxy.

1 29. The system of claim 28 wherein the base agent processing unit allocates an
2 occurrence of an associated base agent corresponding to the page of data.

1 30. The system of claim 29 wherein the object processing unit further calculates output
2 data for the component, and streams out the data to the associated base agent.

1 31. The system of claim 30 wherein the object processing unit further calls a stream
2 result method of the associated base agent.

1 32. The system of claim 30 wherein the object processing unit creates a reference to the
2 associated base agent within the component.

1 33. The system of claim 29 wherein the object processing unit generates at least one
2 container object from the container, and executes the at least one container object,
3 wherein executing comprises.

1 34. The system of claim 33 wherein the object processing unit executes a proxy if the at
2 least one container object is a proxy, executes a component if the at least one
3 container object is a component, and executes a container if the at least one container
4 object is a container.

1 35. The system of claim 28 wherein the base agent processing unit further allocates an
2 occurrence of at least one associated base agent corresponding to the page of data.

1 36. The system of claim 35 wherein the proxy processing unit further determines if a
2 cache entry exists for the occurrence of the at least one object, allocates a new cache
3 entry if a cache entry is not present, and streams out a cache entry value.

1 37. The system of claim 36 wherein the proxy processing unit matches cache criteria for
2 the cache entry, allocates an underlying object associated with the proxy and
3 executes the underlying object if the cache criteria does not match, and constructs a
4 cache key if the cache criteria does match.

1 38. The system of claim 37 wherein the proxy processing unit further matches at least
2 one input parameter against at least one cache criteria entry.

1 39. The system of claim 36 wherein the proxy processing unit further examines the
2 cache using a cache key.

1 40. The system of claim 36 wherein the proxy processing unit further creates the new
2 cache entry in the cache, allocates an occurrence of a caching base agent, and
3 executes the caching base agent.

1 41. The system of claim 40 further comprising a caching agent processing unit to create
2 a buffer entry to capture output data, allocate an underlying object associated with
3 the proxy, execute the underlying object, and transfer the buffer entry to the new
4 cache entry.

1 42. The system of claim 41 wherein the caching agent processing unit executes a proxy
2 if the at least one object is a proxy, executes a component if the at least one object is a
3 component, and executes a container if the at least one object is a container.

1 43. The system of claim 36 wherein the proxy processing unit determines if the cache
2 entry is empty, and streams out an error message if the cache entry is empty.

1 44. The system of claim 28 wherein the at least one object comprises all components
2 within the page of data.

1 45. The system of claim 28 wherein the at least one object is an executable object.

1 46. The system of claim 28 wherein the at least one object is a component, a proxy, or a
2 container.

1

ABSTRACT

A method and system of streaming a page of data are described. In one embodiment, an object corresponding to the page of data is allocated. The object is executed. If the object is a proxy, then the proxy is executed. If the object is a component, then the component is executed. If the object is a container, then the container is executed.

004411.P002

FIG. 1 is a block diagram of a network system 100.

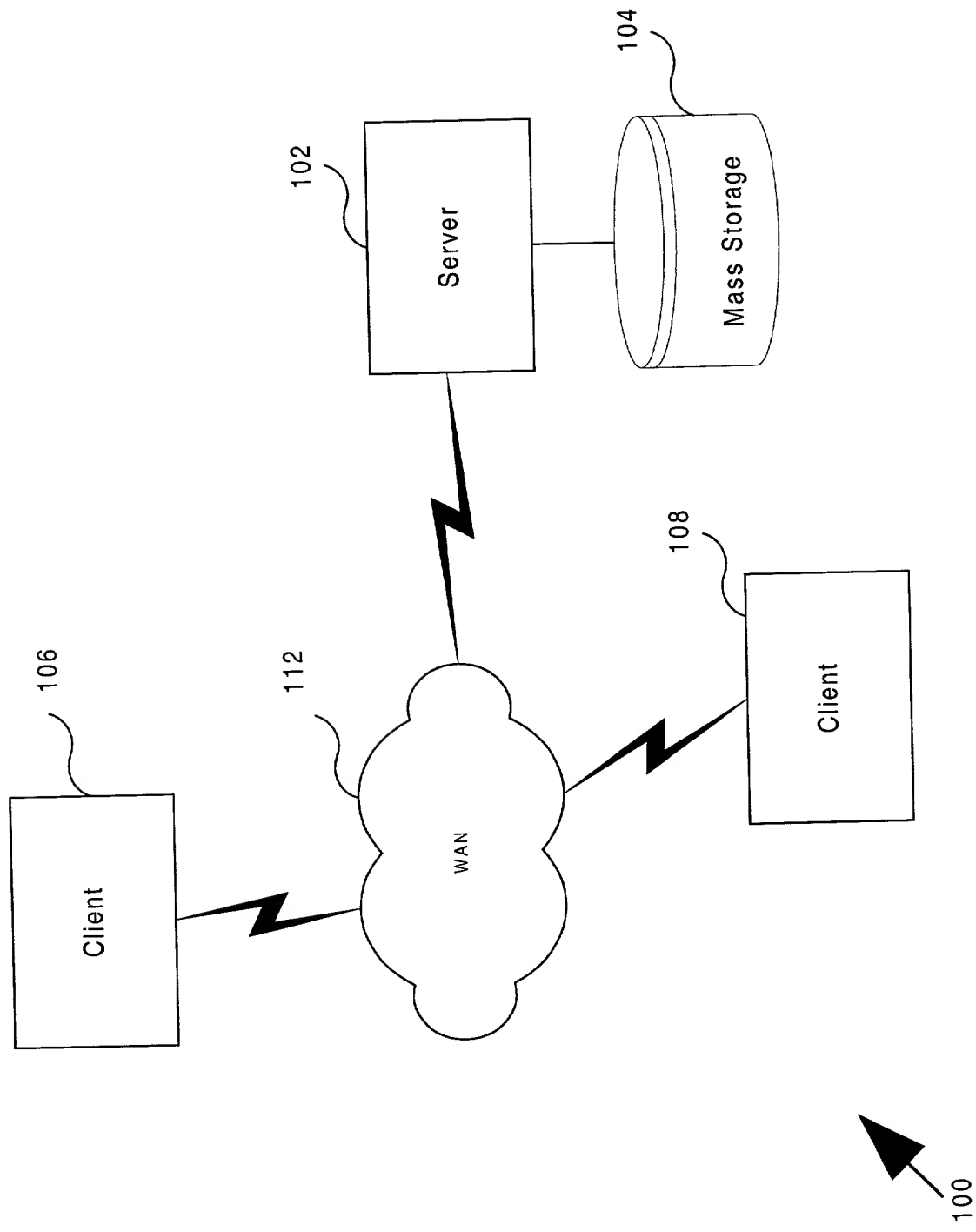


Figure 1

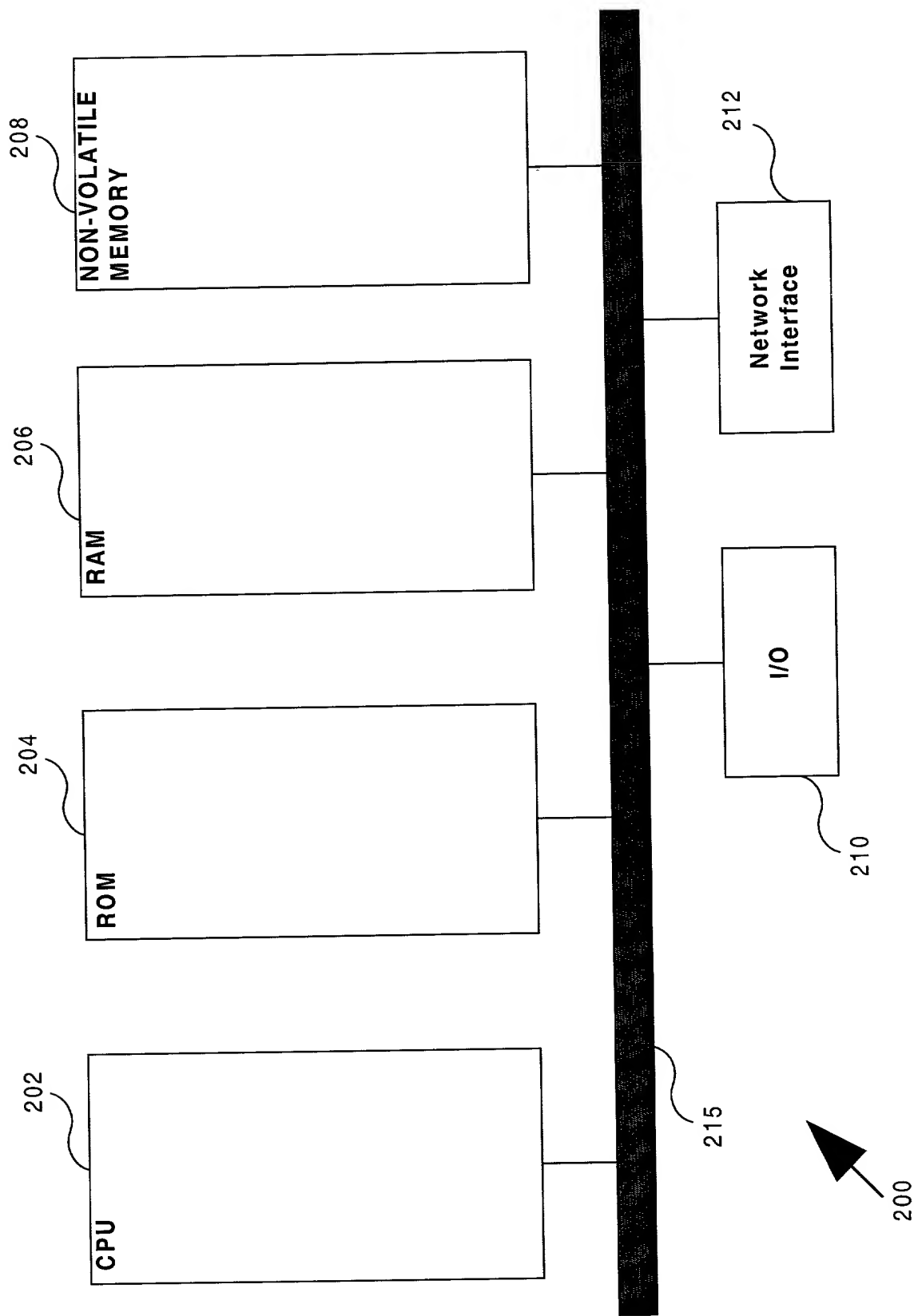


Figure 2

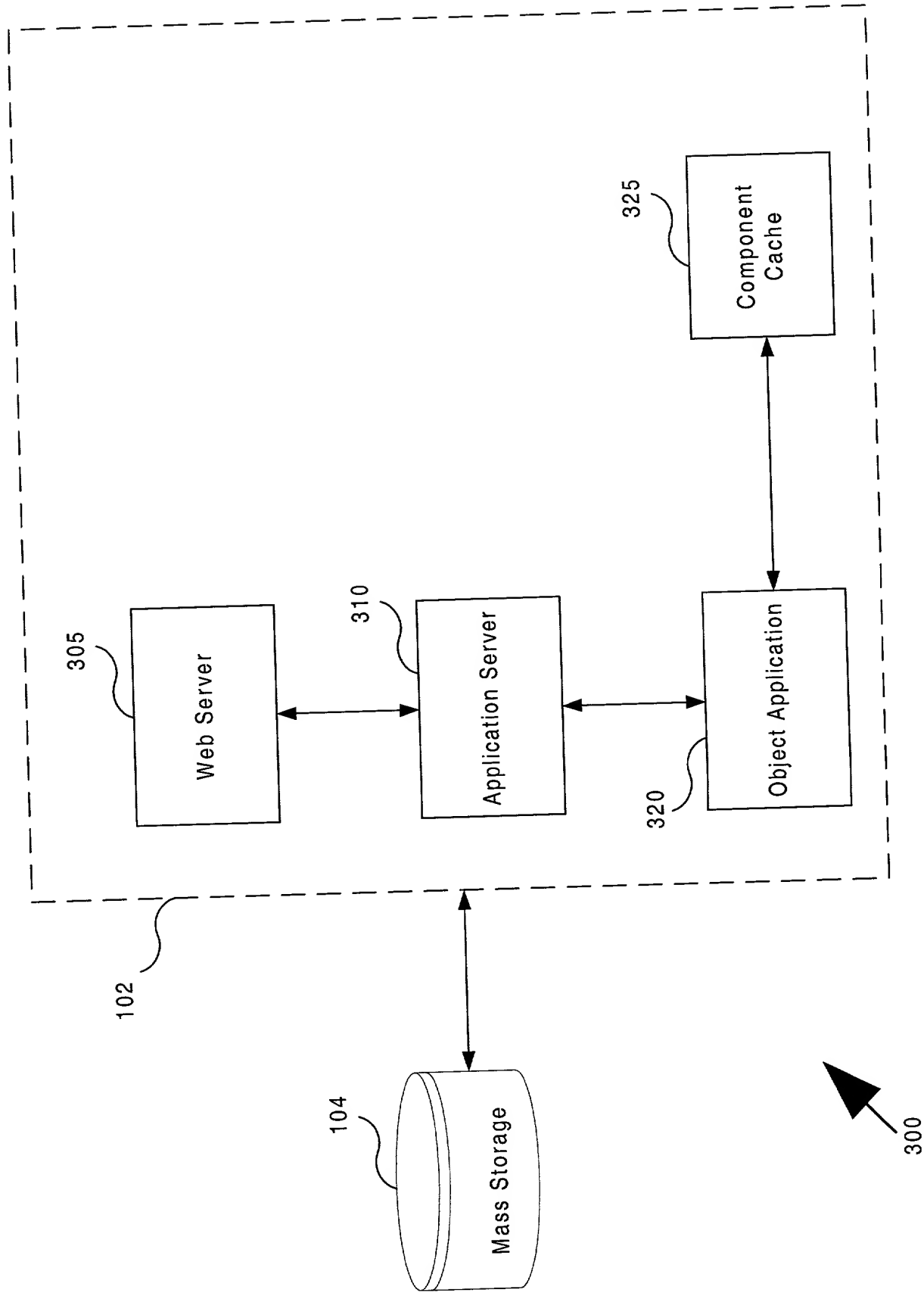


Figure 3

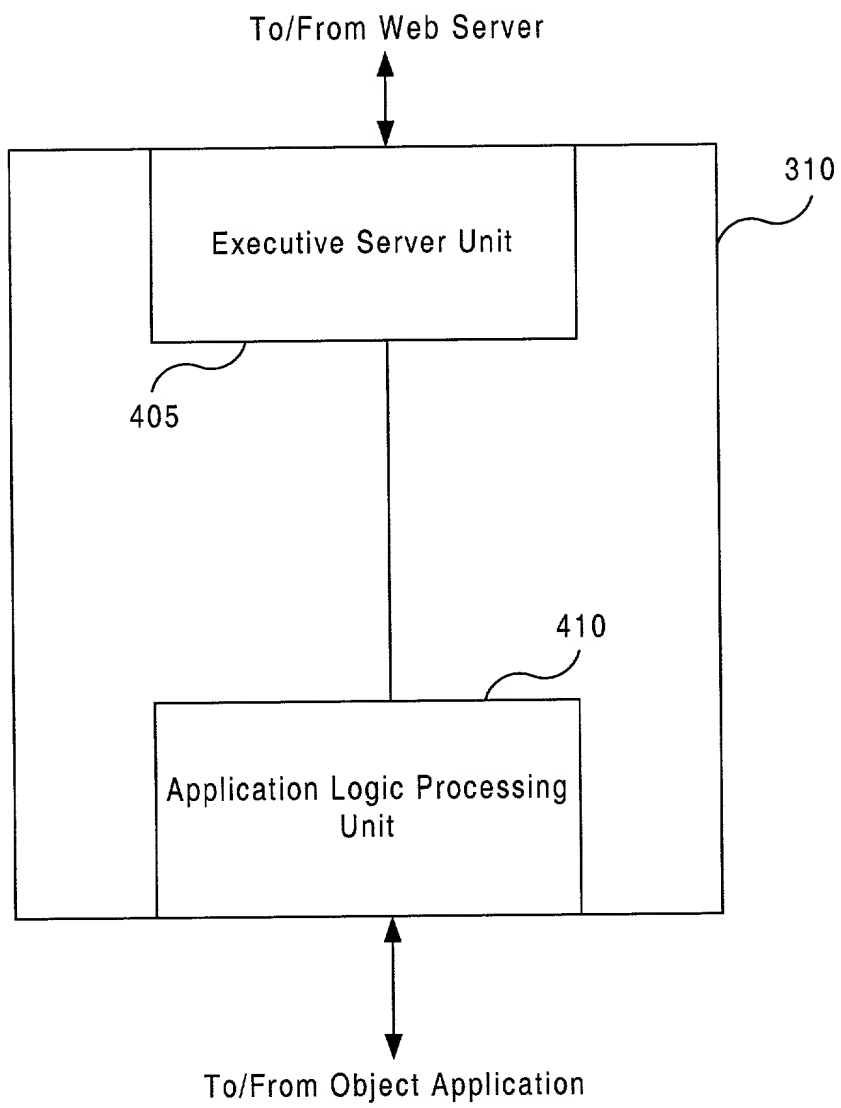


Figure 4a

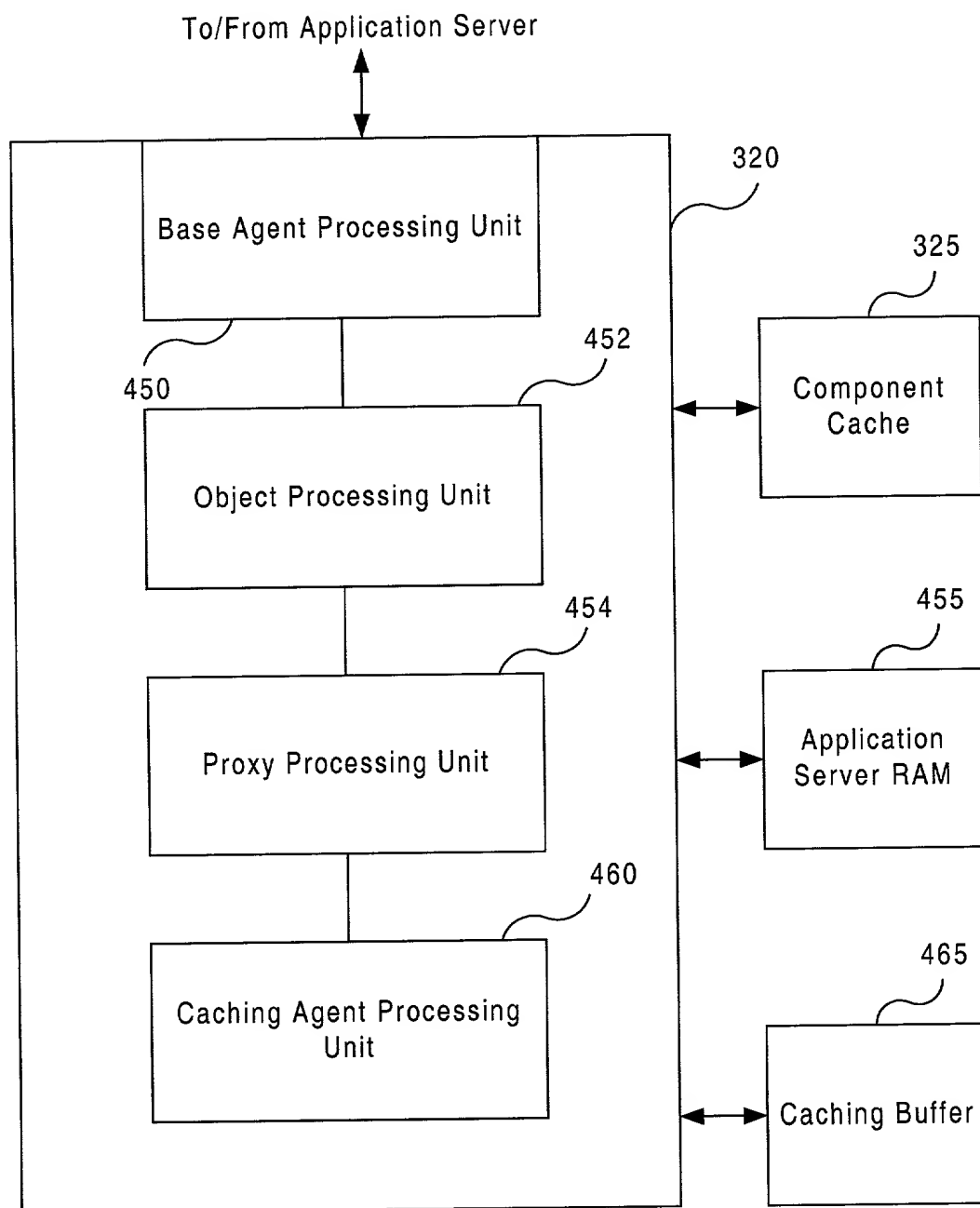


Figure 4b

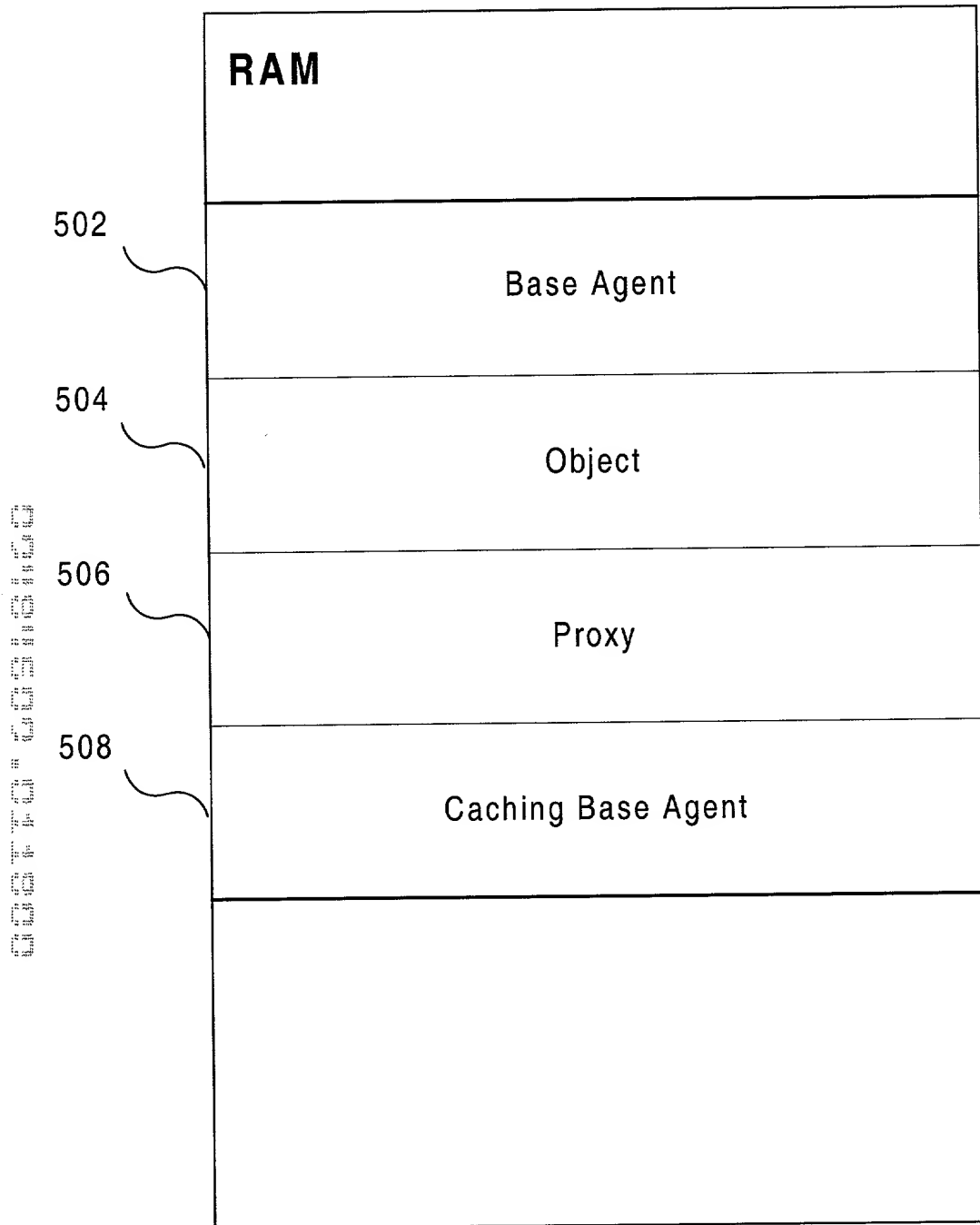


Figure 5

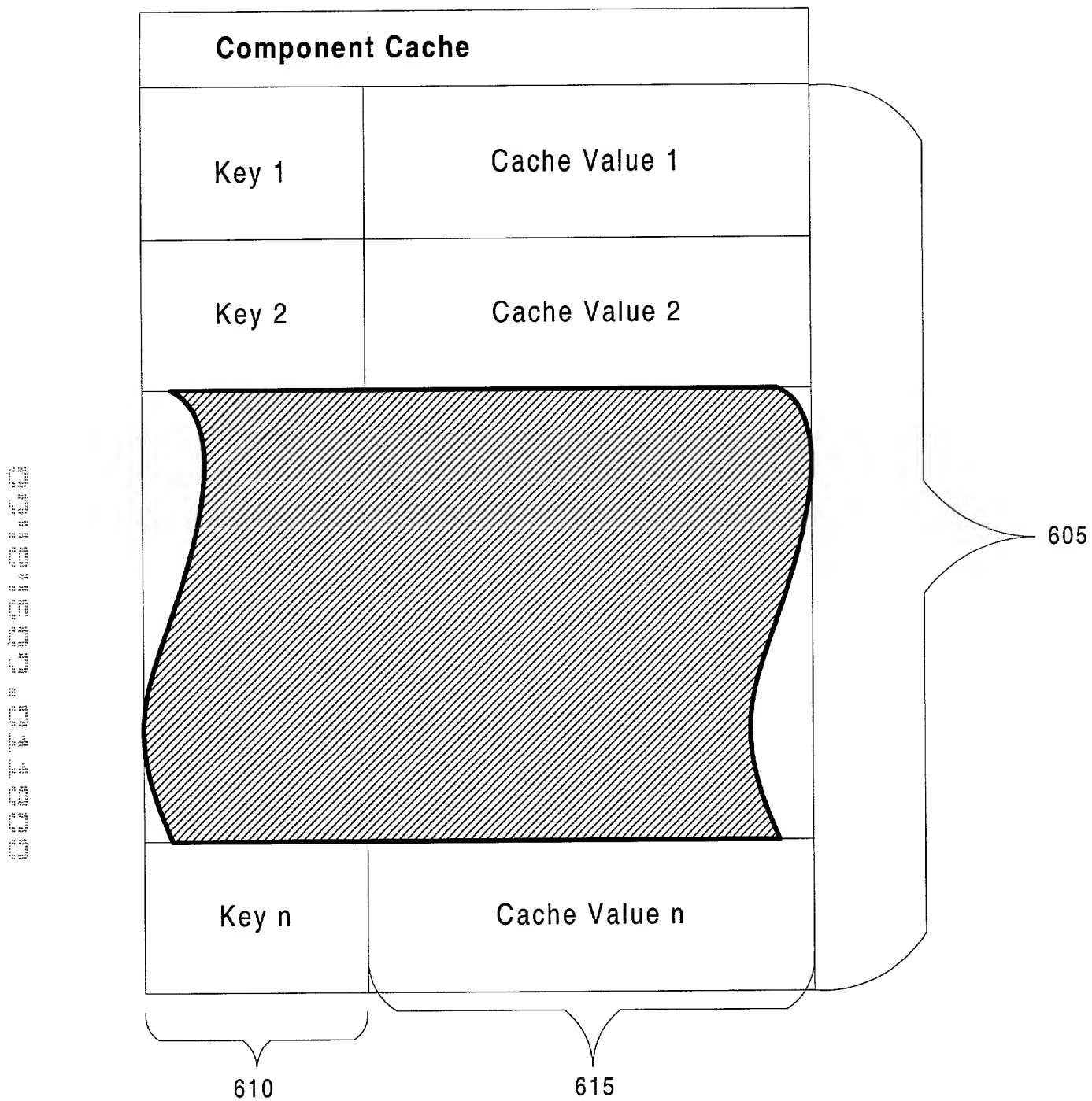


Figure 6

Figure 7

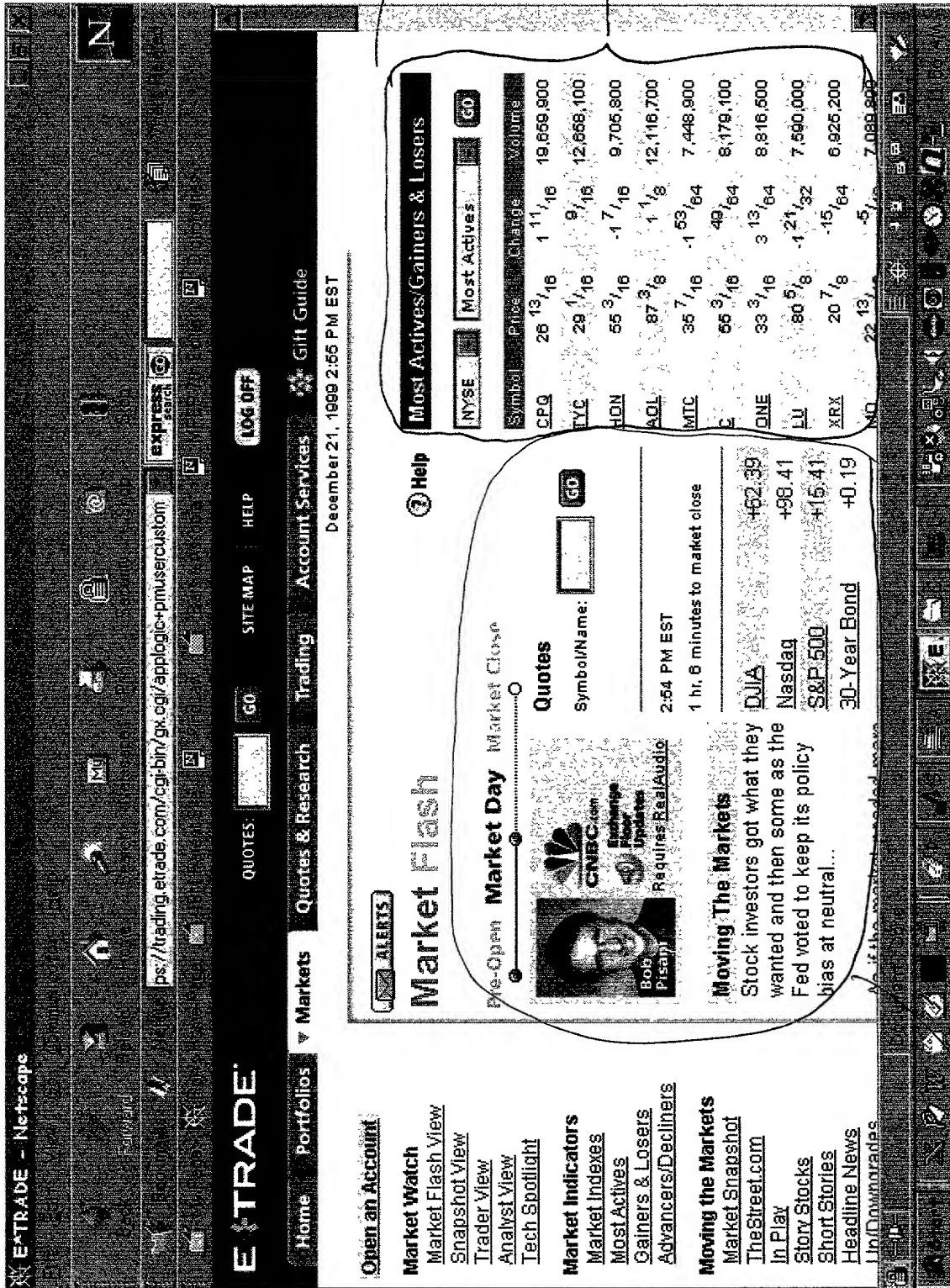


Figure 10

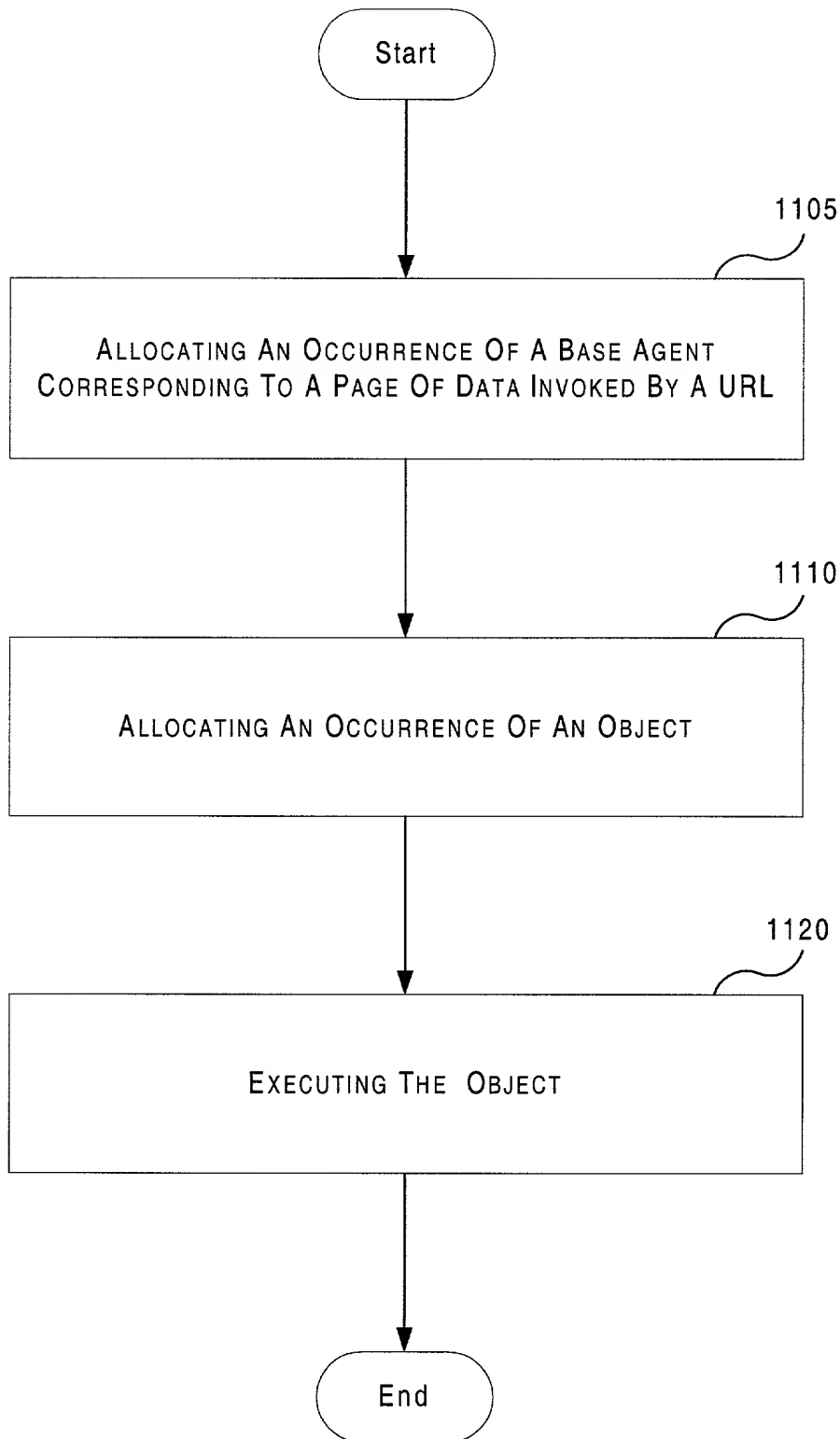


Figure 11

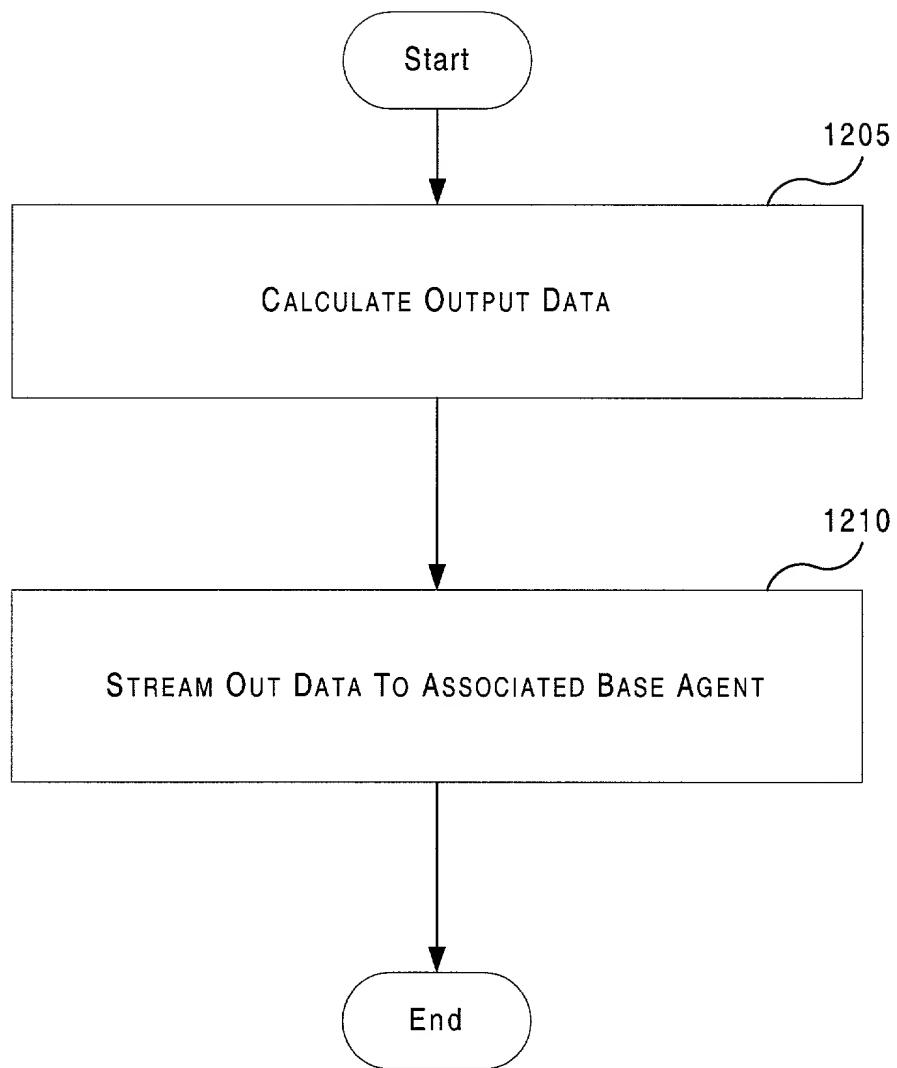


Figure 12

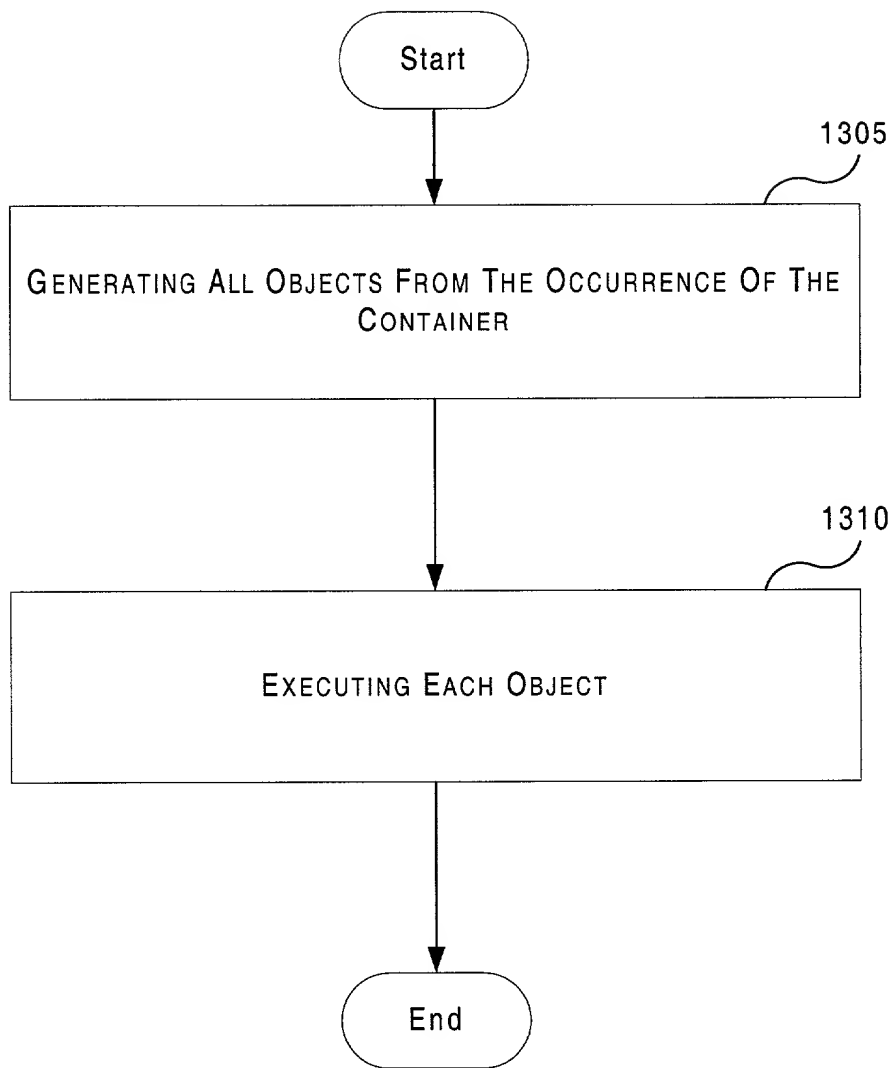


Figure 13

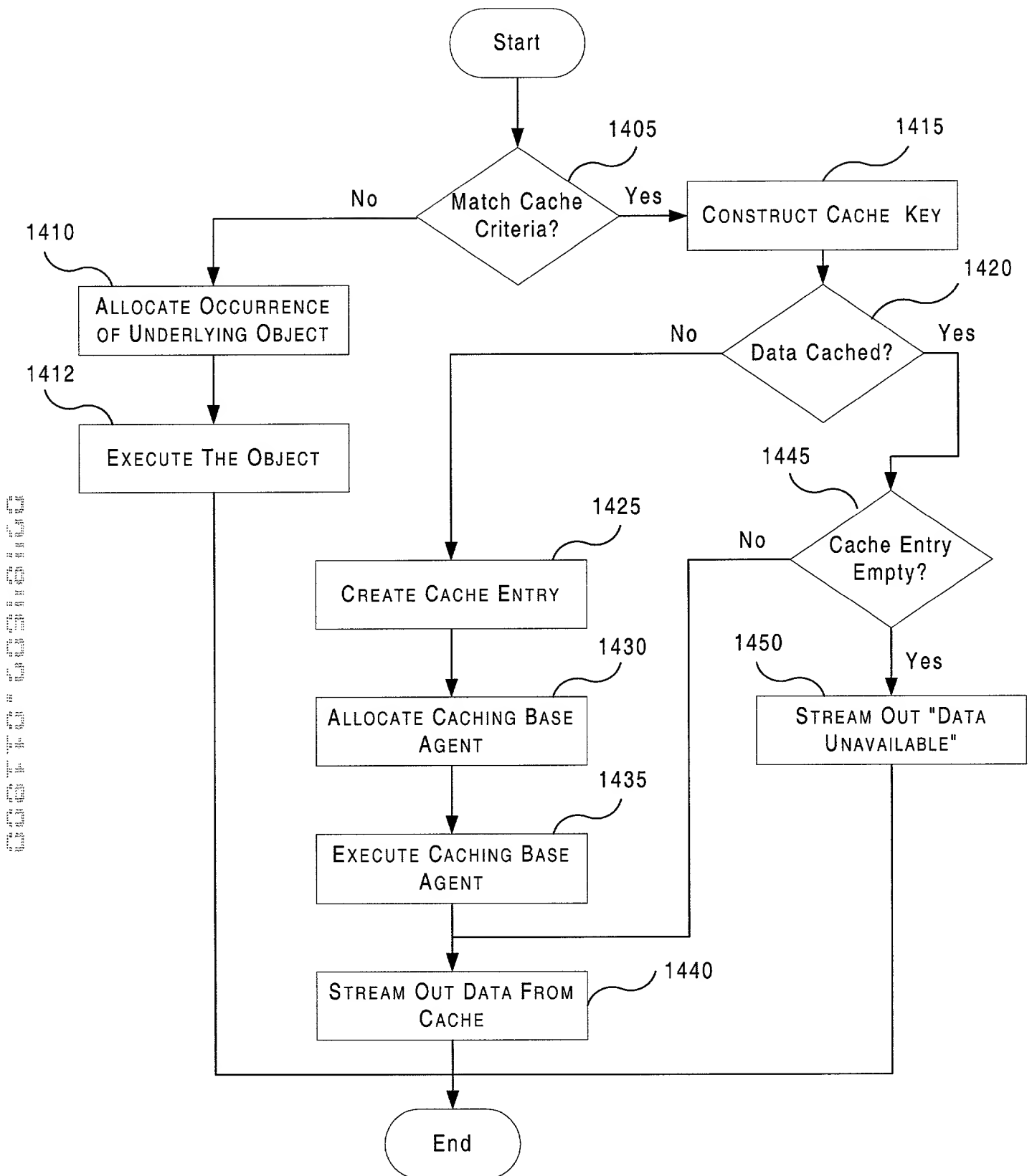


Figure 14

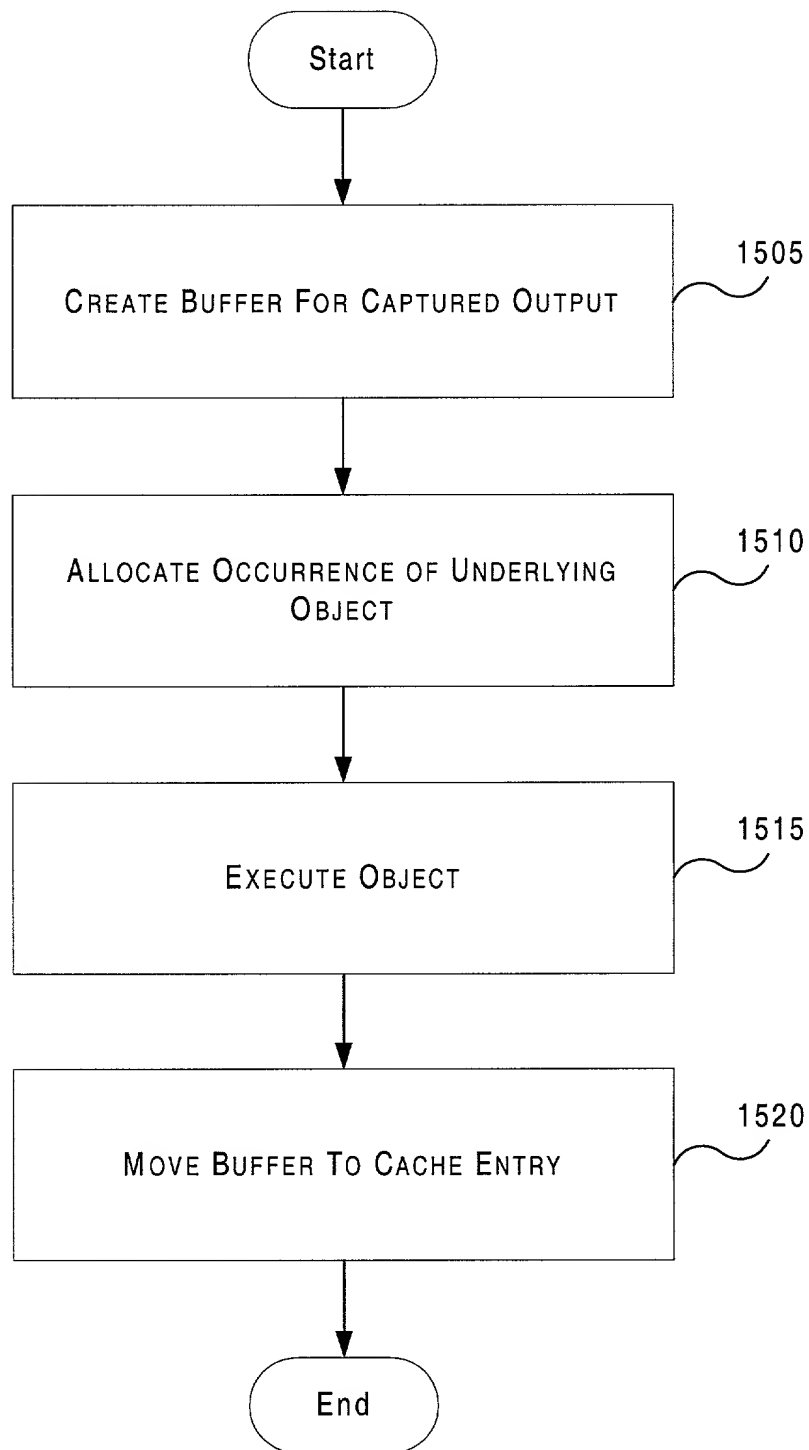


Figure 15

As a below named inventor, I hereby declare that:

My residence, post office address and citizenship are as stated below, next to my name.

I believe I am the original, first, and sole inventor (if only one name is listed below) or an original, first, and joint inventor (if plural names are listed below) of the subject matter which is claimed and for which a patent is sought on the invention entitled
CACHING OUTPUT FROM AN OBJECT IN AN APPLICATION SERVER ENVIRONMENT

the specification of which

X is attached hereto.
_____ was filed on _____ as
United States Application Number _____
or PCT International Application Number _____
and was amended on _____
(if applicable)

I hereby state that I have reviewed and understand the contents of the above-identified specification, including the claim(s), as amended by any amendment referred to above.

I acknowledge the duty to disclose all information known to me to be material to patentability as defined in Title 37, Code of Federal Regulations, Section 1.56.

I hereby claim foreign priority benefits under Title 35, United States Code, Section 119(a)-(d), of any foreign application(s) for patent or inventor's certificate listed below and have also identified below any foreign application for patent or inventor's certificate having a filing date before that of the application on which priority is claimed:

Prior Foreign Application(s)

Priority
Claimed

Number	Country	Day/Month/Year Filed	Yes	No
Number	Country	Day/Month/Year Filed	Yes	No
Number	Country	Day/Month/Year Filed	Yes	No

I hereby claim the benefit under Title 35, United States Code, Section 119(e) of any United States provisional application(s) listed below:

Application Number	Filing Date
Application Number	Filing Date

I hereby claim the benefit under Title 35, United States Code, Section 120 of any United States application(s) listed below and, insofar as the subject matter of each of the claims of this application is not disclosed in the prior United States application in the manner provided by the first paragraph of Title 35, United States Code, Section 112, I acknowledge the duty to disclose all information known to me to be material to patentability as defined in Title 37, Code of Federal Regulations, Section 1.56 which became available between the filing date of the prior application and the national or PCT international filing date of this application:

Application Number	Filing Date	Status -- patented, pending, abandoned

Application Number	Filing Date	Status -- patented, pending, abandoned

I hereby appoint the persons listed on Appendix A hereto (which is incorporated by reference and a part of this document) as my respective patent attorneys and patent agents, with full power of substitution and revocation, to prosecute this application and to transact all business in the Patent and Trademark Office connected herewith.

Send correspondence to Ronald C. Card, BLAKELY, SOKOLOFF, TAYLOR &
(Name of Attorney or Agent)
ZAFMAN LLP, 12400 Wilshire Boulevard 7th Floor, Los Angeles, California 90025 and direct
telephone calls to Ronald C. Card, (408) 720-8598.
(Name of Attorney or Agent)

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

Full Name of Sole/First Inventor Roni Korenshtein

Inventor's Signature _____ Date _____

Residence Los Gatos, CA Citizenship United States of America
(City, State) (Country)

Post Office Address 16407 Shady View Lane
Los Gatos, CA 95032

Full Name of Second/Joint Inventor Rajesh Narayanaswamy

Inventor's Signature _____ Date _____

Residence Sunnyvale, CA Citizenship India
(City, State) (Country)

Post Office Address 655 South Fair Oaks #P212
Sunnyvale, CA 94086

APPENDIX A

William E. Alford, Reg. No. 37,764; Farzad E. Amini, Reg. No. P42,261; Aloysius T. C. AuYeung, Reg. No. 35,432; William Thomas Babbitt, Reg. No. 39,591; Carol F. Barry, Reg. No. 41,600; Jordan Michael Becker, Reg. No. 39,602; Bradley J. Bereznak, Reg. No. 33,474; Michael A. Bernadicou, Reg. No. 35,934; Roger W. Blakely, Jr., Reg. No. 25,831; Gregory D. Caldwell, Reg. No. 39,926; Ronald C. Card, Reg. No. 44,587; Thomas M. Coester, Reg. No. 39,637; Stephen M. De Klerk, under 37 C.F.R. § 10.9(b); Michael Anthony DeSanctis, Reg. No. 39,957; Daniel M. De Vos, Reg. No. 37,813; Robert Andrew Diehl, Reg. No. 40,992; Matthew C. Fagan, Reg. No. 37,542; Tarek N. Fahmi, Reg. No. 41,402; James Y. Go, Reg. No. 40,621; James A. Henry, Reg. No. 41,064; Willmore F. Holbrow III, Reg. No. P41,845; Sheryl Sue Holloway, Reg. No. 37,850; George W. Hoover II, Reg. No. 32,992; Eric S. Hyman, Reg. No. 30,139; Dag H. Johansen, Reg. No. 36,172; William W. Kidd, Reg. No. 31,772; Erica W. Kuo, Reg. No. 42,775; Michael J. Mallie, Reg. No. 36,591; Andre L. Marais, under 37 C.F.R. § 10.9(b); Paul A. Mendonsa, Reg. No. 42,879; Darren J. Milliken, Reg. No. 42,004; Lisa A. Norris, Reg. No. P44,976; Chun M. Ng, Reg. No. 36,878; Thien T. Nguyen, Reg. No. 43,835; Thinh V. Nguyen, Reg. No. 42,034; Dennis A. Nicholls, Reg. No. 42,036; Kimberley G. Nobles, Reg. No. 38,255; Daniel E. Ovanezian, Reg. No. 41,236; Babak Redjaian, Reg. No. 42,096; William F. Ryann, Reg. No. 44,313; James H. Salter, Reg. No. 35,668; William W. Schaal, Reg. No. 39,018; James C. Scheller, Reg. No. 31,195; Jeffrey Sam Smith, Reg. No. 39,377; Maria McCormack Sobrino, Reg. No. 31,639; Stanley W. Sokoloff, Reg. No. 25,128; Judith A. Szepesi, Reg. No. 39,393; Vincent P. Tassinari, Reg. No. 42,179; Edwin H. Taylor, Reg. No. 25,129; John F. Travis, Reg. No. 43,203; George G. C. Tseng, Reg. No. 41,355; Joseph A. Twarowski, Reg. No. 42,191; Lester J. Vincent, Reg. No. 31,460; Glenn E. Von Tersch, Reg. No. 41,364; John Patrick Ward, Reg. No. 40,216; Charles T. J. Weigell, Reg. No. 43,398; Kirk D. Williams, Reg. No. 42,229; James M. Wu, Reg. No. P45,241; Steven D. Yates, Reg. No. 42,242; Ben J. Yorks, Reg. No. 33,609; and Norman Zafman, Reg. No. 26,250; my patent attorneys, and Andrew C. Chen, Reg. No. 43,544; Justin M. Dillon, Reg. No. 42,486; Paramita Ghosh, Reg. No. 42,806; and Sang Hui Kim, Reg. No. 40,450; my patent agents, of BLAKELY, SOKOLOFF, TAYLOR & ZAFMAN LLP, with offices located at 12400 Wilshire Boulevard, 7th Floor, Los Angeles, California 90025, telephone (310) 207-3800, and James R. Thein, Reg. No. 31,710, my patent attorney.

APPENDIX B

Title 37, Code of Federal Regulations, Section 1.56 Duty to Disclose Information Material to Patentability

(a) A patent by its very nature is affected with a public interest. The public interest is best served, and the most effective patent examination occurs when, at the time an application is being examined, the Office is aware of and evaluates the teachings of all information material to patentability. Each individual associated with the filing and prosecution of a patent application has a duty of candor and good faith in dealing with the Office, which includes a duty to disclose to the Office all information known to that individual to be material to patentability as defined in this section. The duty to disclose information exists with respect to each pending claim until the claim is cancelled or withdrawn from consideration, or the application becomes abandoned. Information material to the patentability of a claim that is cancelled or withdrawn from consideration need not be submitted if the information is not material to the patentability of any claim remaining under consideration in the application. There is no duty to submit information which is not material to the patentability of any existing claim. The duty to disclose all information known to be material to patentability is deemed to be satisfied if all information known to be material to patentability of any claim issued in a patent was cited by the Office or submitted to the Office in the manner prescribed by §§1.97(b)-(d) and 1.98. However, no patent will be granted on an application in connection with which fraud on the Office was practiced or attempted or the duty of disclosure was violated through bad faith or intentional misconduct. The Office encourages applicants to carefully examine:

(1) Prior art cited in search reports of a foreign patent office in a counterpart application, and

(2) The closest information over which individuals associated with the filing or prosecution of a patent application believe any pending claim patentably defines, to make sure that any material information contained therein is disclosed to the Office.

(b) Under this section, information is material to patentability when it is not cumulative to information already of record or being made of record in the application, and

(1) It establishes, by itself or in combination with other information, a prima facie case of unpatentability of a claim; or

(2) It refutes, or is inconsistent with, a position the applicant takes in:

(i) Opposing an argument of unpatentability relied on by the Office, or

(ii) Asserting an argument of patentability.

A prima facie case of unpatentability is established when the information compels a conclusion that a claim is unpatentable under the preponderance of evidence, burden-of-proof standard, giving each term in the claim its broadest reasonable construction consistent with the specification, and before any consideration is given to evidence which may be submitted in an attempt to establish a contrary conclusion of patentability.

(c) Individuals associated with the filing or prosecution of a patent application within the meaning of this section are:

(1) Each inventor named in the application;

(2) Each attorney or agent who prepares or prosecutes the application; and

(3) Every other person who is substantively involved in the preparation or prosecution of the application and who is associated with the inventor, with the assignee or with anyone to whom there is an obligation to assign the application.

(d) Individuals other than the attorney, agent or inventor may comply with this section by disclosing information to the attorney, agent, or inventor.